Требования к программам

1. Программа работает с массивом объектов типа `student`:

```
enum class io_status
{
  success,
  eof,
  format,
  memory,
};

class student
{
  private:
    char * name  = nullptr;
    int    value = 0;
  public:
    student () = default;
    student (const student& x) = delete;
    student (student&& x)
      {
        name = x.name; x.name = nullptr;
        value = x.value; x.value = 0;
      }
    ~student ()
      {
        erase ();
      }
student& operator= (const student& x) = delete;
student& operator= (student&& x)
  {
    if (this == &x)
      return *this;
    erase ();
    name = x.name; x.name = nullptr;
    value = x.value; x.value = 0;
    return *this;
  }
void print () const
  {
    printf ("%s %d\n", name, value);
  }
io_status read (FILE * fp)
  {
    const int LEN = 1234;
    char n[LEN];
    int v;

    if (fscanf (fp, "%s%d", n, &v) != 2)
      {
        if (! feof (fp))
```

```
            return io_status::format;
          return io_status::eof;
        }
      erase();
      return init (n, v);
    }
  int operator> (const student& x) const
    {
      return (cmp (x) > 0 ? 1 : 0);
    }
  int operator< (const student& x) const
    {
      return (cmp (x) < 0 ? 1 : 0);
    }
  int operator== (const student& x) const
    {
      return (cmp (x) == 0 ? 1 : 0);
    }

private:
  io_status init (const char * n, int v)
    {
      value = v;
      if (n != nullptr)
        {
          size_t len = strlen (n);
          name = new char [len + 1];
          if (name != nullptr)
            {
              for (size_t i = 0; i <= len; i++)
                name[i] = n[i];
            }
          else
            return io_status::memory;
        }
      return io_status::success;
    }
void erase ()
  {
    value = 0;
    if (name != nullptr)
      {
        delete [] name;
        name = nullptr;
      }
  }
int cmp (const student& x) const
  {
    if (name == nullptr)
      {
        if (x.name != nullptr)
```

```
            return -1;
          return value - x.value;
        }
      if (x.name == nullptr)
        return 1;
      int res = strcmp (name, x.name);
      if (res)
        return res;
      return value - x.value;
    }
};
```

2. Программа должна получать все параметры в качестве аргументов командной строки.

3. Аргументы командной строки **для задач 1–5**:

   1) n – размерность массива,
   2) p – количество выводимых значений в массиве,
   3) s – задает номер формулы для инициализации массива, должен быть равен 0 при вводе массива из файла,
   4) `filename` – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s! = 0$.

   Например, запуск

   ```
   ./a04.out 4 4 0 a.txt
   ```

   означает, что массив длины 4 надо прочитать из файла `a.txt`, и выводить не более 4-х элементов массива, а запуск

   ```
   ./a04.out 1000000 6 1
   ```

   означает, что массив длины 1000000 надо инициализировать по формуле номер 1, и выводить не более 6-ти элементов массива.

4. Аргументы командной строки **для задач 6–10**:

   1,2) x – дополнительный аргумент (x – объект типа `student`, передаваемый как пара: строка и целое число),
   3) n – размерность массива,
   4) p – количество выводимых значений в массиве,
   5) s – задает номер формулы для инициализации массива, должен быть равен 0 при вводе массива из файла,
   6) `filename` – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s! = 0$.

5. Ввод массива должен быть оформлен в виде подпрограммы, находящейся в отдельном файле.

6. Ввод массива из файла. В указанном файле находится массив в формате:

$$\begin{array}{ll} \text{Слово-1} & \text{Целое-число-1} \\ \text{Слово-2} & \text{Целое-число-2} \\ \dots & \dots \\ \text{Слово-n} & \text{Целое-число-n} \end{array}$$

   где n - указанный размер массива, слово – последовательность алфавитно-цифровых символов без пробелов. Программа должна выводить сообщение об ошибке, если указанный файл не может быть прочитан, содержит меньшее количество данных или данные неверного формата.

7. Ввод массива по формуле. Элемент a_i массива A полагается равным

$$a_i = \{\text{"Student"}, f(s,n,i)\}, \quad i = 1,\dots,n, \tag{1}$$

   где $f(s,n,i)$ - функция, которая возвращает значение (i)-го элемента массива по формуле номер s (аргумент командной строки). Функция $f(s,n,i)$ должна быть оформлена в виде отдельной подпрограммы `int f (int s, int n, int i);` работающей только с целыми числами.

$$f(s,n,i) = \begin{cases} i & \text{при} \quad s = 1 \\ n-i & \text{при} \quad s = 2 \\ i/2 & \text{при} \quad s = 3 \\ n-i/2 & \text{при} \quad s = 4 \end{cases}$$

   Инициализация объекта по формуле (1) должна быть оформлена как член класса `student`.

8. Решение задачи должно быть оформлено в виде подпрограммы, находящейся в отдельном файле и получающей в качестве аргументов массив и его длину (в 6 – 10 задачах также дополнительные аргументы). Получать в этой подпрограмме дополнительную информацию извне через глобальные переменные, включаемые файлы и т.п. запрещается. Также в этой функции запрещается что-либо выводить на экран.

9. Программа должна содержать подпрограмму вывода на экран массива длины не более p. Эта подпрограмма используется для вывода исходного массива после его инициализации, а также для вывода на экран результата. Подпрограмма выводит на экран не более, чем p элементов массива, где p – параметр этой подпрограммы (аргумент командной строки). Каждый элемент массива должен печататься на новой строке.

10. Результатом работы каждой функции является измененный массив (выводится в `main`) и возвращаемое значение – его новая длина (выводится в `main`).

11. Вывод результата работы функции в функции `main` должен производиться по формату:

```
printf ("%s : Task = %d Result = %d Elapsed = %.2f\n",
        argv[0], task, res, t);
```

    где

    - `argv[0]` – первый аргумент командной строки (имя образа программы),
    - `task` – номер задачи (1–10),
    - `res` – возвращаемое значение функции, реализующей решение этой задачи,
    - `t` – время работы функции, реализующей решение этой задачи.

    Вывод должен производиться в точности в таком формате, чтобы можно было автоматизировать обработку запуска многих тестов.

Задачи

1. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и выбрасывающую из массива элементы, меньшие предыдущего; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

2. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и выбрасывающую из массива элементы, большие 2-х предыдущих; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

3. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и выбрасывающую из массива элементы, меньшие соседних элементов; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

4. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и выбрасывающую из массива элементы, большие 2-х соседних элементов слева и справа; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

5. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и выбрасывающую из массива все участки убывания; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

6. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и ссылку на объект типа `student` x, и выбрасывающую из массива все элементы, меньшие x; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

7. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и ссылку на объект типа `student` x, и выбрасывающую из массива все подряд идущие элементы, меньшие x, оставляя только первый элемент в таком участке; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

8. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и ссылку на объект типа `student` x, и выбрасывающую из массива все подряд идущие элементы, большие x, оставляя только первый и последний элементы в таких участках; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

9. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и ссылку на объект типа `student` x, и выбрасывающую из массива все элементы, имеющие соседа (слева или справа), меньшего x; функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).

10. Написать функцию, получающую в качестве аргументов массив $a[n]$ объектов типа `student`, целое число n, являющееся длиной этого массива, и ссылку на объект типа `student` x, и выбрасывающую из массива все участки возрастания, целиком состоящие из элементов, меньших x, функция возвращает длину получившегося в результате массива (число пересылок элементов массива не должно превосходить $n+O(1)$).